Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Science and Technology Houari Boumediene

Project Report

Development of a Multiple-Choice Questionnaire (MCQ) Application

Presented by : **LES SCHTROUMPFETTES**

•Kiheli Ahlem

•Benyettou Hadil

•Khouas Ihsene Noor

•Bentoumi Lyna Racha

3ème année ING sécurité
ANNÉE UNIVERSITAIRE : 2024/2025

Table of contents:

1. Introduction

The objective of this project is to develop a Python-based application for administering Multiple-Choice Questionnaires (MCQs). Designed for computer science students, the application provides a platform for answering questions, tracking scores, and receiving personalized feedback. Additionally, the system includes user management features and records performance histories. The application's console-based interface ensures ease of use and broad compatibility.

2. Technical Choices

2.1 Programming Language: Python

Python was chosen for its simplicity, extensive library support, and suitability for rapid development. Its versatility allows seamless integration of JSON for data management, enhancing the application's functionality and maintainability.

2.2 Data Storage

JSON Structure

The application utilizes JSON files to store user information, question data, and quiz results. This choice ensures:

- Lightweight data storage.
- Human-readable format for easy debugging and updates.
- Support for structured data through dictionaries and lists.

User Data

Each user entry contains the following:

- Username
- Role (e.g., admin or student)
- Password
- Quiz history, including dates, scores, categories, and time taken.

Example:

```
{
   "username": "user1",
   "role": "student",
   "password": "password123",
   "history": [
      {
         "date": "2025-01-04",
         "score": 10,
         "category": "Algorithms",
```

```
        "time_taken": 19.59
      }
    ]
}
```

Questions

Questions are categorized and stored in JSON files. Each entry includes:

- Question text
- Options
- Correct answer index
- Category

Example:

```
{
   "category": "Python",
   "question": "What is the correct file extension for Python files?",
   "options": [".py", ".python", ".pyt"],
   "correct": 0
}
```

2.3 User Interface

The application features a console-based interface structured as follows:

1. User Management: Users log in or register, ensuring personalized experiences.
2. Category Selection: Users select question categories.
3. Questionnaire: Questions are presented one at a time, with feedback provided for each answer.
4. Result Display: A summary of performance, including score and time taken, is shown.

3. Features

3.1 User Management

- Registration: New users create accounts with unique usernames and passwords.
- Authentication: Existing users log in to access their history.
- History Tracking: Past quiz performances, including scores and timestamps, are displayed.

3.2 Question Handling

- Questions are retrieved dynamically based on the chosen category.

- Feedback includes both correct answers and explanations for mistakes.

3.3 Time Management

A time limit is imposed on quizzes:

- Per Question Timer: Limits user response time for each question.
- Overall Timer: A maximum duration for completing the quiz.

3.4 Result Export

Scores and histories can be exported to CSV format for offline analysis.

3.5 Advanced Features

1. Categories: Questions are organized by topics like Python, Networking, and Algorithms.
2. Custom Timer: Total and per-question timers improve user focus.
3. Feedback Mechanism: Correct answers and explanations help users improve.

4. Challenges and Solutions

4.1 Input Validation

Challenge: Handling invalid inputs from users.

- Solution: Implement loops and validation mechanisms to prompt for correct inputs.

4.2 Data Synchronization

Challenge: Maintaining real-time updates to user histories.

- Solution: Save data immediately after each update to prevent data loss.

4.3 Timer Implementation

Challenge: Accurate measurement of time limits.

- Solution: Use Python's `time` module to track elapsed time and enforce limits.

5. Implementation Details

Core Files:

- `qcm_app.py`: Contains the main logic, including user authentication, question display, and score calculation.
- `questions.json`: Stores question data, categorized by topic.

- `users.json`: Stores user profiles and history.

Functions

User Management

- `load_users()`: Loads user data from JSON.
- `save_users()`: Updates the JSON file with new user data.
- `register_user()`: Registers new users.
- `authenticate_user()`: Validates existing users.

Question Management

- `load_questions()`: Loads questions from JSON.
- `choose_category()`: Allows users to select a category.
- `display_questions()`: Presents questions, calculates scores, and tracks time.

Time Handling

- `format_time()`: Converts seconds into `MM:SS` format for better readability.

6. Conclusion

This project demonstrates the effective use of Python for developing an educational application. By integrating features like real-time feedback, timer constraints, and advanced user management, the application offers a robust tool for learning. Future improvements could include:

- A graphical interface for enhanced usability.
- Integration of a question editor for admins.
- Advanced analytics on user performance.